Unsupervised dense crowd detection by multiscale texture analysis

Fagette Antoine, Nicolas Courty, Daniel Racoceanu, Jean-Yves Dufour

▶ **To cite this version:**

Fagette Antoine, Nicolas Courty, Daniel Racoceanu, Jean-Yves Dufour. Unsupervised dense crowd detection by multiscale texture analysis. Pattern Recognition Letters, 2013, pp.1-27. hal-00904210

HAL Id: hal-00904210
https://hal.science/hal-00904210

Submitted on 14 Nov 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Unsupervised Dense Crowd Detection by Multiscale Texture Analysis

Antoine Fagette[a,d], Nicolas Courty[b], Daniel Racoceanu[c,d], Jean-Yves Dufour[e]

[a] *Thales Solutions Asia Pte Ltd*
28 Changi North Rise - Singapore 498755, Singapore
antoine.fagette@thalesgroup.com
Tel: +65 8200 8957 - Fax: +65 6478 9920
[b] *IRISA - Université de Bretagne Sud*
Campus de Tohannic - 56000 Vannes, France
[c] *University Pierre and Marie Curie*
4 Place Jussieu, 75005 Paris, France
[d] *CNRS, IPAL UMI 2955*
1 Fusionopolis Way, #10-18 Connexis (South Tower) - Singapore 138632, Singapore
[e] *Thales Services, Campus Polytechnique*
1 Avenue Augustin Fresnel - 91767 Palaiseau Cedex, France

Abstract

This study introduces a totally unsupervised method for the detection and location of dense crowds in images without context-awareness. With the perspective of setting up fully autonomous video-surveillance systems, automatic detection and location of crowds is a crucial step that is going to point which areas of the image have to be analyzed. After retrieving multiscale texture-related feature vectors from the image, a binary classification is conducted to determine which parts of the image belong to the crowd and which to the background. The algorithm presented can be operated on images without any prior knowledge of any kind and is totally unsupervised.

Keywords: dense crowd, segmentation, feature extraction, texture analysis, quadtree, diffusion maps, multiscale

1. Introduction

Crowd monitoring has become a major concern of the beginning of the 21st century. With the increasing number of CCTV networks in public areas, the enhancement of the computing power of modern computers and the progress made these past decades in computer sciences and computer vision in particular, the possibility to entrust an automatic system with the security and the monitoring of events involving large crowds is within reach.

This paper is dealing with the problem of detection and location of a dense crowd in the image. It focuses on a method that does not need any training set nor any prior knowledge of any kind on the context from where the picture or the video has been taken. Our method is based on the assumption that a crowd is visually identified by a type of texture characterized by great variations of the color vectors as well as of the orientations of the borders. The features that we extract from the image are representative of these variations. We are also taking into account the multiscale aspect of a crowd by appending several feature vectors computed with several sizes of spatial neighborhood, thus forming a multiscale feature vector. Unlike the previous studies on this same topic, detailed in Section 2, our work, described in Section 3, aims at providing a method that is totally unsupervised and independent of the shooting conditions. It is based on the appearance of the crowd and not exclusively on its motion. We are therefore able to locate dynamic as well as static crowds on images taken from cameras and poses we know nothing of. We have run tests on static images of both synthetic and real scenes within which the ground truth is known. The results of this experimentation,

detailed in Section 4, prove that our method is successful at detecting dense crowds.

2. State of the art

When it comes to crowd detection, several methods have been developed, each valid for its own context, mainly depending on the density of the crowd to be detected and its distance to the camera. The goal of this Section is to give a brief overview of the different methods that are used for the detection and the location of crowds in video-surveillance.

Through different surveys on crowd analysis in general ([1] and [2]) and on pedestrian detection and human visual analysis in particular ([3] and [4]), crowd detection can be generalized into three main ways: to detect the pedestrians themselves, to proceed with background substraction methods and/or to assume that in the observed scene every moving object is part of a crowd.

The process of recognizing each pedestrian of a crowd to detect assumes that, in the image, it is possible to segment each pedestrian from the background or from a group of pedestrians. It requires from the camera to be close enough to have a number of pixels per pedestrian high enough to run the algorithm but has the advantage to work theoretically well with still images as well as with mobile cameras. Nevertheless, this method reaches its limits when the crowd is too dense and the number of occlusions too important for the algorithm to match its human model with the objects it detects. It may also fail when the relative motion between the camera and the pedestrians is too chaotic to enable a good capture of the phenomenon.

The pedestrian recognition is used by Leibe *et al.* in [5] where they combine local and global cues via a probabilistic top-down segmentation to identify the human beings. Wu and Nevatia in [6] use edgelet features to segment pedestrians even partially occluded, and so do Lin *et al.* in [7] by generating a body part template to match as well as possible the detected shapes among the crowd. Dalal and Triggs, in [8], prove the efficiency of the Histograms of Oriented Gradients to detect a pedestrian in the image. Finally, Tu *et al.* in [9] detect heads and shoulders as a first guess on the positions of the pedestrians and then associate every squared sub-part of the image to the most probable pedestrian or to the background.

Another way of proceeding is to use a background substraction algorithm. This method goes with the assumption that each object that is not part of the background is going to be a pedestrian or that an algorithm is able, afterwards, to classify the detected objects as pedestrian or non-pedestrian (in the latter, the techniques are quite close to those described in the previous paragraph). This technique is not able to deal with video streams taken from a mobile camera. However, it is very efficient to monitor places such as pedestrian zones, stadiums or fairs where the environment is well controlled and only pedestrians are expected. Dong in [10] manages to detect human beings even with some occlusions by matching the shapes detected from the background substraction with models and combined models. Wang and Yung in [11] match 3D human models with silhouette obtained via background substraction and helps to find the best position for his models by locating the heads of the pedestrians through a head detector.

The third method that is commonly used for crowd detection assumes

that a crowd is never static and that it evolves in an environment that is, itself, non-dynamic. Therefore, by using for example an optical flow algorithm, one can detect the areas of the image where something is moving and deduct the position of the crowd. This process finds its limits when the camera itself is moving (beyond possible correction) or in the case when the crowd is standing still (*e.g.* sit-ins, commemorations, etc.). Boghossian and Velastin in [12] use an optical flow algorithm to get the motion, introduce continuity properties to remove the noise and detect slow movements by running this optical flow algorithm between two frames separated by several others. Reisman *et al.* in [13] use an optical flow algorithm as well and detect the movements that can only be made by a human crowd with specific classifiers, thus eliminating the vehicles. Rabaud and Belongie in [14] use both the motion and the fact that a crowd is composed of objects that are similar in shape to locate the crowd and its pedestrians. Finally, Ali and Shah in [15] use a set of particles combined with an optical flow algorithm to detect the flow and to go further by identifying its instabilities.

Recently, some work has been done to detect and locate a crowd in the image using texture analysis. Indeed, a dense crowd has a very particular aspect, made of a patchwork of colors, that lead researchers to consider this feature to segment a crowd from the background. This is precisely the idea that is exploited by Manfredi *et al.* in [16] to detect and locate groups of pedestrians in open spaces, using classification. In [17], Rodriguez *et al.* combine a head detector, belonging to the first technique described above, to the results given by density estimation to robustify their crowd detection and pedestrian location.

The method described in this paper belongs to this last category of methods. Following Manfredi *et al.*, we use a texture-based approach. However, our priority is given to the unsupervised learning to separate the crowd from the background.

3. Overview of the method

Our method aims at detecting large dense crowds in which it is impossible to segment each individual and without any training dataset. It is based on a texture analysis technique. First, from each pixel of the image, features relevant to the crowd texture are extracted. These features are stored in a vector of features attached to the described pixel. Then the pixels are classified either as belonging to the crowd or to the background. This binary classification is performed using a diffusion map with the extracted features as data. This last operation raises a problem of time and volume of computation that leads us to consider reducing the amount of data to be treated.

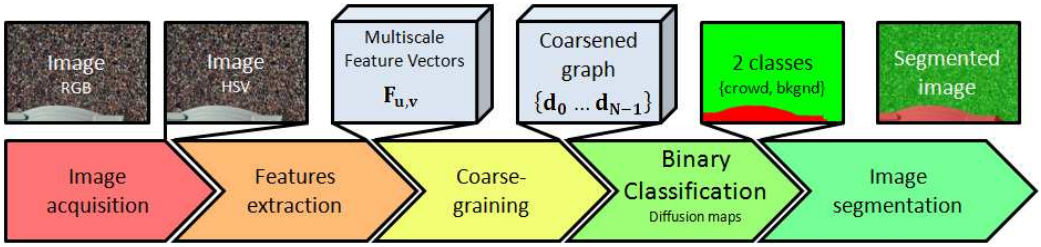


Figure 1: Overview of the method

3.1. Features extraction

We base our work on the assumptions that the color vectors describing a crowd show great spatial variations and that the borders of objects (namely

the pedestrians) are orientated in all possible directions. These assumptions are to be opposed to the one assuming that the background tends to be more uniform on wider areas. Retrieving features representing the level of variation of the color vector and of the orientation of the borders at each pixel of the image gives us a good level of information on the presence or not of the crowd on this pixel.

3.1.1. Window of observation dimensioning

The study deals with images taken by non-calibrated cameras. Therefore, the size of the sample from which the features are extracted cannot be determined uniquely. This problem is avoided by considering that each size of this window of observation gives a different value of the same feature.

Therefore, if we suppose that we extract n features using m sizes of window of observation $\{r_1, \ldots, r_m\}$, then, at every pixel (u, v) of the image I, the value of the i^{th} feature for the j^{th} size of window of observation r_j is $f_{u,v}^{i,r_j}$ and we obtain the multiscale feature vector $\mathbf{F_{u,v}}$:

$$\forall (u,v) \in I,\ \mathbf{F_{u,v}} = \begin{pmatrix} f_{u,v}^{1,r_1} \\ \vdots \\ f_{u,v}^{1,r_m} \\ \vdots \\ f_{u,v}^{n,r_1} \\ \vdots \\ f_{u,v}^{n,r_m} \end{pmatrix} \tag{1}$$

3.1.2. Features definition

Three types of features are implemented for this study: the Laplacian of Gaussian (LoG), the entropy and the Histogram of Oriented Gradients

(HOG). As we want to use the information carried by the color itself, we choose to work within the HSV colorspace. We use the hue component I_h in radians to compute the LoG and the entropy and weight each of these two features with the saturation component I_s. The HOG is computed with the value component I_v.

Therefore, for each pixel (u, v) of the image I and for each size of window of observation r_j, the computation of these three features gives respectively $f_{u,v}^{1,r_j}$, $f_{u,v}^{2,r_j}$ and $f_{u,v}^{3,r_j}$. Because of the angular nature of the terms of I_h, we use their complex values $\tilde{I}_h$ and the smallest angular difference $\Delta_{\theta_1}^{\theta_2}$ for the computation of the LoG:

$$\forall (u,v) \in I_h,\ \tilde{I}_h(u,v) = \exp\left(i \cdot I_h(u,v)\right) \tag{2}$$

$$\Delta_{\theta_1}^{\theta_2} = (\theta_2 - \theta_1 + \pi) \bmod (2\pi) - \pi \tag{3}$$

In the following, $\otimes$ denotes a term-by-term multiplication, $*$ a convolution: $\forall (u,v) \in I$,

$$f_{u,v}^{1,r_j} = \left(G_{\sigma_j} * LoG_I\right)(u,v) \tag{4}$$

$$f_{u,v}^{2,r_j} = \left(-\sum_{k=0}^{b} \frac{G_{\sigma_j} * B_k \otimes log_2\left(G_{\sigma_j} * B_k\right)}{log_2(N)} \right) \otimes \left(G_{\sigma_j} * I_s\right)^{\beta}(u,v) \tag{5}$$

$$f_{u,v}^{3,r_j} = \|\mathbf{f}_{\mathbf{u},\mathbf{v}}^{\mathbf{3},\mathbf{r_j}}\| \tag{6}$$

with LoG_I the customized LoG:

$$LoG_I(u,v) = \sum_{U=u-r_j}^{u+r_j} \sum_{V=v-r_j}^{v+r_j} \Delta_{arg\left(\left(G_{\sigma_1} * \tilde{I}_h\right)(U,V)\right)}^{arg\left(\left(G_{\sigma_1} * \tilde{I}_h\right)(u,v)\right)} \cdot \left(I_s(u,v) \cdot I_s(U,V)\right)^{\alpha} \tag{7}$$

and G_{σ_j} and G_{σ_1}, the normalized gaussian filters defined respectively by $\sigma_j = \frac{r_j}{3}$ and $\sigma_1 = \frac{1}{3}$.

B_k is the binary image corresponding to the k^{th} bin of the histogram of b bins used to compute the entropy:

$$\forall (u,v) \in I_h,\ B_k(u,v) = \begin{cases} 1 & \text{if } \frac{2k\pi}{b} \leq I_h(u,v) < \frac{2(k+1)\pi}{b}, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

and $\mathbf{f}_{\mathbf{u},\mathbf{v}}^{\mathbf{3},\mathbf{r_j}}$ is the result of the convolution of each bin of the HOG performed on I_v by the gaussian filter G_{σ_j} at pixel (u,v). It is a vector of dimension d if the orientation is binned into d bins. Usually, and it is our case, $d = 8$. For more details on the HOG, the reader may refer to [8] by Dalal and Triggs.

With the experience, we choose: $\alpha = \beta = 0.25$.

In the end, we obtain, for each pixel (u,v) of the image, a multiscale feature vector of dimension $3 \cdot m$. This multiscale feature vector is then normalized in order to ensure the coherence of the data.

3.2. Clustering using a diffusion map and classification

Once the multiscale feature vectors have been computed, pixels have to be labeled as the crowd or the background. As we are focusing on an unsupervised method we are looking for a clustering algorithm that could separate the pixels according to the value of their attached multiscale feature vectors. However, as opposed to traditional methods, such as K-means, that are often considering only the distance between two data to determine whether they belong or not to the same cluster, we want to use both the lengths and the density of the different paths between these data, as suggested by Farbman *et al.* in [19]. This can be achieved using a spectral clustering.

It is also reasonable to assume that the different densities observed in the scene with different illumination conditions will lead to different feature

vectors that will nevertheless belong to the same manifold. The diffusion maps, as initially introduced by Coifman and Lafon in [20], are an interesting tool that preserves the similarity between those samples while providing a low-dimensional embedding which encodes the structural information of the manifold. Beside the spectral clustering aspect, the diffusion maps algorithm is also embedding a parameter, the diffusion parameter, hereafter noted t, that can be seen as a scaling parameter. Scrolling this parameter from one value to another can strengthen or weaken the relationship existing between two data points. The diffusion maps algorithm is therefore used to divide the multiscale feature vectors in two clusters.

A good introduction to the diffusion maps can be read in the paper of de la Porte *et al.* [21]. The idea of using this approach for clustering is described by Nadler *et al.* in [22]. We base our work with the diffusion maps on these papers, using a gaussian kernel to map the multiscale feature vectors in the diffusion space.

Then, for both clusters, the mean vector of all the attached multiscale feature vectors is computed. The one with the highest norm gets the crowd label, the other the background one.

The diffusion maps technique is a powerful tool yet subject to some limitations regarding the amount of data to be processed. Using the algorithm directly on the multiscale feature vectors associated to each pixel of an image with a 4CIF resolution implies clustering 405504 elements. The diffusion matrix holds therefore more than 160 billion values and the complexity is skyrocketing. The amount of data to be processed has to be limited in order to reduce the time and volume of computation.

This problem has been addressed in various ways. Among them, an approach described by Fowlkes *et al.* in [23] and used by Farbman *et al.* in [19] is based on the Nyström method. It approximates the eigenvalues and eigenvectors of the diffusion matrix using a smaller sample of the data, taken randomly. Afterwards, it computes the missing points using the Nyström extension. Another method, developed by Lafon and Lee in [24] is, with the same idea of sub-sampling, to regroup data that are similar into clusters and to build the diffusion map with these clusters and no longer with the data themselves. The coarse-grained version of the original diffusion map is supposed to have the same spectral properties provided that the choice of the clusters has been made correctly.

Our approach is different in the sense that it tries to coarsen the graph while considering the spatial relationship between the elements of the graph. It is based on the computation of a quadtree.

3.3. Quadtree computation

The difficulty with quadtrees lies into finding the criterion that will indicate if a region of rank k contains data that are homogeneous enough or else if it needs to be split into four sub-regions of rank $k+1$.

In our case, the data used for each sub-region is the mean vector of all the multiscale feature vectors attached to the pixels contained in this sub-region. We note M_{0_i} the i^{th} region of rank k, M_{1_i}, M_{2_i}, M_{3_i} and M_{4_i} its four sub-regions of rank $k+1$ and $\mathbf{m_{0_i}}$, $\mathbf{m_{1_i}}$, $\mathbf{m_{2_i}}$, $\mathbf{m_{3_i}}$ and $\mathbf{m_{4_i}}$ their respective data. The level of homogeneity H is then evaluated using $\mathbf{V_{0_i}}$, the variance vector of the four data $\mathbf{m_{1_i}}$, $\mathbf{m_{2_i}}$, $\mathbf{m_{3_i}}$ and $\mathbf{m_{4_i}}$ of the sub-regions, and $\mathbf{V_I}$

the variance vector of all the multiscale feature vectors of the image.

$$\mathbf{V_{0_i}} = Var\left(\{\mathbf{m_{1_i}}, \mathbf{m_{2_i}}, \mathbf{m_{3_i}}, \mathbf{m_{4_i}}\}\right) \tag{9}$$

$$\mathbf{V_I} = Var\left(\left\{\mathbf{F_{u,v}}\right\}_{(u,v)\in I}\right) \tag{10}$$

$$H = \begin{cases} True & \text{if } \forall l \in \{0, \cdots, 3\cdot m - 1\}, \mathbf{V_{0_i}}[l] < \alpha \cdot \mathbf{V_I}[l] \\ False & \text{otherwise.} \end{cases} \tag{11}$$

With α a parameter set by the user. With the experience, we choose α between 0% and 20%. If H is false, the region M_{0_i} is considered as not homogeneous, it is split into the four sub-regions M_{1_i}, M_{2_i}, M_{3_i} and M_{4_i} which level of homogeneity is going to be tested at the next iteration of k. If H is true, the region M_{0_i} is considered as homogeneous, it will not be further split and becomes a leaf of our quadtree. If M_{0_i} is the j^{th} leaf of the quadtree we note:

$$L_j = M_{0_i} \tag{12}$$

$$\mathbf{d_j} = \mathbf{m_{0_i}} \tag{13}$$

In the end, the quadtree is composed of N leaves $\{L_0, \cdots, L_{N-1}\}$ with their respective data $\{\mathbf{d_0}, \cdots, \mathbf{d_{N-1}}\}$. These are the data used to compute the diffusion map and perform the binary classification described in 3.2.

4. Validation

For the validation, two datasets of images[1] are used. The first one is composed of images that have been retrieved from Google Images and manually

[1]These datasets are available for download at http://www.ipal.cnrs.fr/download

annotated. These images show large dense crowds in urban environments, presenting challenging backgrounds with textures ranging from near-regular to near-stochastic. Their resolution is diverse and they are all JPEG-encoded. The second dataset was generated with the synthetic crowd generator Agoraset built by Allain *et al.* and described in [26]. This second dataset is automatically annotated by the crowd generator. The annotation for both datasets is the following: green represents the crowd, red the background. The same colors are used to display our results.

In Section 3 three parameters of the algorithm have been declared:

- m, the number of sizes of window of observation and the value of these sizes (see Subsection 3.1)
- α, the quadtree parameter for the homogeneity (see Subsection 3.3)
- t, the diffusion parameter of the diffusion maps algorithm (see Subsection 3.2)

In this Section, we give the conclusions of a series of tests that study the influence of each of these parameters on the performances of the algorithm.

Then, taking advantage of the capability of the crowd generator to produce different textures, we challenged our algorithm into finding a crowd on different backgrounds. The results are illustrated and commented in this Section.

We compare also the efficiency of our algorithm to the efficiency of the traditional K-means.

Finally, we provide some of the results produced by our method as well as an evaluation of its performances on two different sets of images issued

from our datasets.

To evaluate the performance of our algorithm we use the F-score indicator and assign Positive to the crowd class and Negative to the background class.

4.1. Influence of m

The parameter m is embedding two characteristics: the number of windows of observation m itself and the sizes of these windows of observation $\{r_1, \cdots, r_m\}$. We have therefore conducted two tests here. For these two tests, we fixed $\alpha = 10\%$ and $t = 1$.

First, we chose five intervals of values ($r_1 = 1$ and $r_m = 5$, 10, 20, 50 or 100) with a unit step between each values, $r_{i+1} - r_i = 1$. The results confirmed the intuition that the range covered by the number of windows of observation has a positive impact on the results given by the algorithm. The wider the range, the better the results. However, in order to have a reasonable size for our multiscale feature vectors, we limit the range covered from $r_1 = 1$ to $r_m = 50$.

In the second test, we fixed $r_1 = 1$ and $r_m = 50$ but we took five different values for the step: 1, 2, 3, 5 and 10. The results showed that the value of the step has no significant influence on the performance, provided that it allows a good sampling of the range r_1 to r_m.

From the results of these two tests, we choose to keep $m = 5$ so that $r_1 = 1$ and $r_{i+1} - r_i = 10$.

4.2. Influence of α

The parameter α determines the level of subdivision of our quadtree, therefore the number of leaves and so the complexity of the diffusion maps

algorithm that is run afterwards. For this set of tests, α took successively the following values: 0%, 5%, 10%, 15% and 20%. We fixed $t = 1$ and $m = 5$ so that $r_1 = 1$ and $r_{i+1} - r_i = 10$. As expected, the results showed that the smaller α is, the more precise the segmentation is but often at a level that is not wished. For scenes with a low complexity, α has little influence however, when the complexity grows, it is preferable to have α not too small in order to avoid an over-segmentation.

Moreover, the parameter α is a bargain parameter. We are trading precision in order to enhance the speed of the diffusion maps part of the algorithm. A balance has to be found and from the tests we have conducted, we choose to keep α between 5% and 10%. It is hard to quantify the time that is saved as a function of α because the level of subdivision of the tree depends on the complexity of the image itself. However, we have set the smallest size a leaf can take to 5 pixels. Below, the information of homogeneity does not make sense any more. Therefore, should the quadtree go to its maximum allowed subdivision, for a 4CIF image, it would have 4096 leaves, *i.e.* 4096 data to be treated by the diffusion maps algorithm to be compared to the 405504 pixels that would have to be clustered if the coarse-graining part was skipped. It represents almost a hundred times less data, therefore the number of operation is divided by 10^6.

4.3. Influence of t

The diffusion parameter t rules the proximity between the data in the diffusion space and has an influence on the final K-means clustering performed in the diffusion space. For this batch of tests, we have fixed $\alpha = 10\%$ and $m = 5$ so that $r_1 = 1$ and $r_{i+1} - r_i = 10$ and t took the following values: 1,

2, 3, 5, 10, 20 and 100.

The results showed that the parameter t gives optimum results for $t = 1$. For t greater than 1, the diffusion map algorithm tends to bring the data too close from each other in the diffusion space, which leads to a bad separation of the data. That is the reason why we choose to keep $t = 1$.

4.4. Influence of the background

In order to test the robustness of our detection method under different background conditions, we used the synthetic ground truth of Figure 2a to test various background conditions. We chose a progression in the complexity of the texture ranging from quasi-flat to quasi-noise, by gradually increasing the level of noise while downgrading the geometric structure of the texture. The first image Figure 2b corresponds to a flat color background and, as such, is the simplest. In the second image Figure 2c, the scene is illuminated with a global illumination model based on photon maps producing shading effects on the ground. In Figure 2d, to the same illumination model is added a virtual sun which casts shadows over the ground. Next, volumetric textures are used with various levels of geometric structures. First a marble texture is used on Figure 2e, then a Brownian noise on Figure 2f, and finally a Markov Random Field over the 3 color components on Figure 2g. This last image is assumed to be the most challenging for our algorithm.

As expected, the performances are quite good on the first three images. The algorithm achieves the detection of the crowd. However, it does not perform a complete detection of the isolated pedestrians, thus downgrading the F-score. The result displayed on Figure 2d shows that the algorithm is sensitive to the shadows but classifies those belonging to the pedestrians

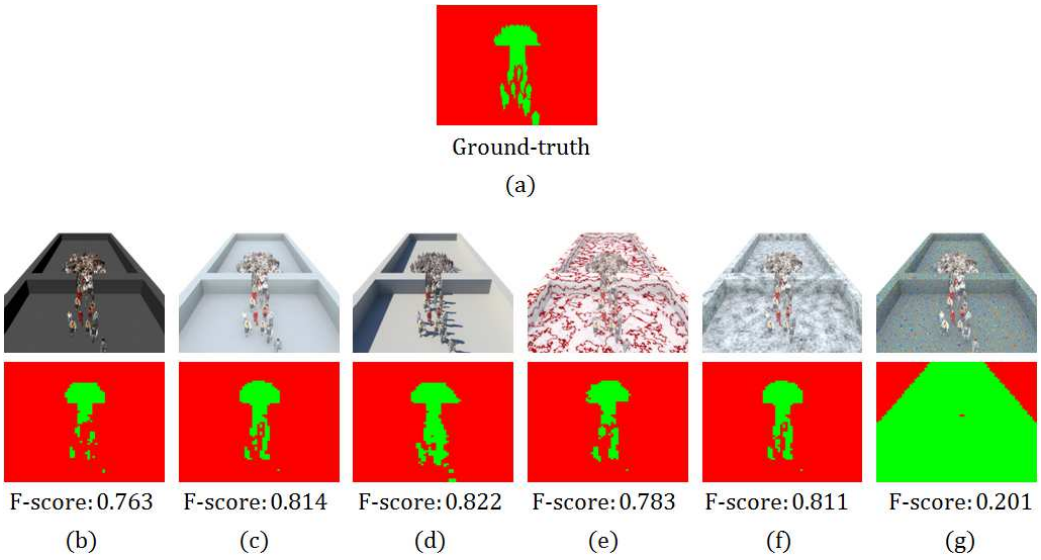


Figure 2: Comparison of the results obtained on a same image of crowd but with various textures for the background.

into the crowd and those belonging to the walls as part of the background. On the marble and Brownian noise backgrounds, the algorithm proves to be very efficient with results comparable to those obtained on the simpler backgrounds.

Finally, our algorithm is out-challenged by the Markov Random Field background. Due to the type of features used by the algorithm, this last result was expected.

4.5. Comparison with the traditional K-means

In this Subsection we compare the performances of the K-means algorithm with the performances of our algorithm. We used the K-means algorithm to separate the m-dimension space containing the multiscale feature vectors into

two clusters. For this set of tests we use four images, two taken from Google Images and two synthetic.

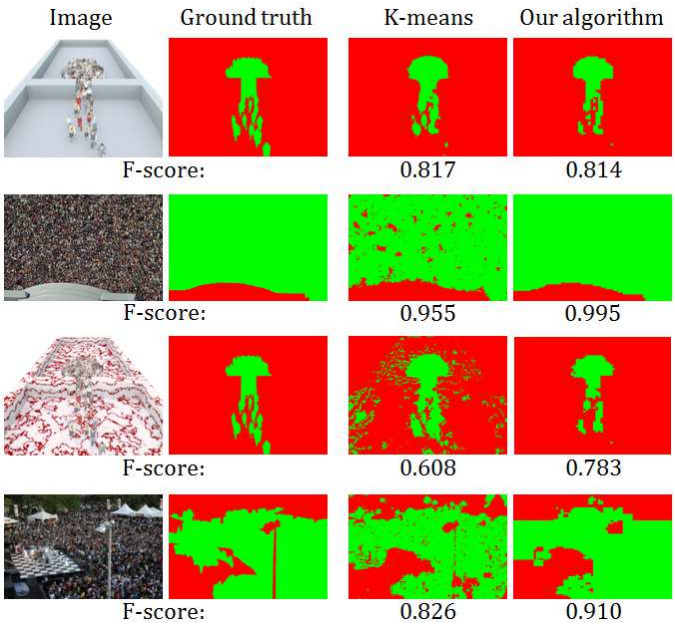


Figure 3: Comparison of the results obtained via a K-Means algorithm with those given by our method.

The results, displayed on Figure 3, show that the performances of the K-means algorithm are comparable with those of our algorithm for scenes of a lower complexity. However, as the complexity grows, the performances of the K-means algorithm decrease significantly whereas those of our algorithm remain higher.

Moreover, the K-means algorithm considers all the pixels independently from each others. This leads to the non-regularity of the two classes (especially for the crowd class). Our algorithm avoids this problem which gives results closer to the human perception.

4.6. More results and performances

This Subsection is providing more results and performances of our algorithm on two sets of images. The first set is composed of ten images synthesized by Agoraset, the second has ten images from Google Images.

In the results that we are providing on Figure 4, one can see that our algorithm comes with good performances and detects efficiently the crowds on the various images that have been used. The F-score computed for the images from the synthetic dataset indicates that our results are less accurate on those images than on the ones from the Google Images dataset. This can be explained by two factors. First, the ground truth for the synthetic dataset is computed by the simulator itself which segments very precisely each pedestrian. On the dataset taken from the Internet, the ground truth has been annotated manually and is therefore subject to the simplifications a human-being tends to make naturally. Since our algorithm works with a quadtree, it mimics this behavior. The second reason that explains the lower performances on the synthetic dataset is that this dataset contains images with isolated pedestrians. Even though the algorithm achieves to detect these pedestrians, it fails most of the time to detect them entirely causing the F-score to drop down.

We would like also to emphasize the problem of the subjectivity inherent to the definition of a crowd. It is indeed debatable until what extent a group of human being can be considered as a crowd or as part of it. On an image, are the inter-individual spaces part of the crowds, or should they be considered as part of the background? Furthermore, are the persons sitting at the terrace of a cafe part of the crowd gathered on the street right in

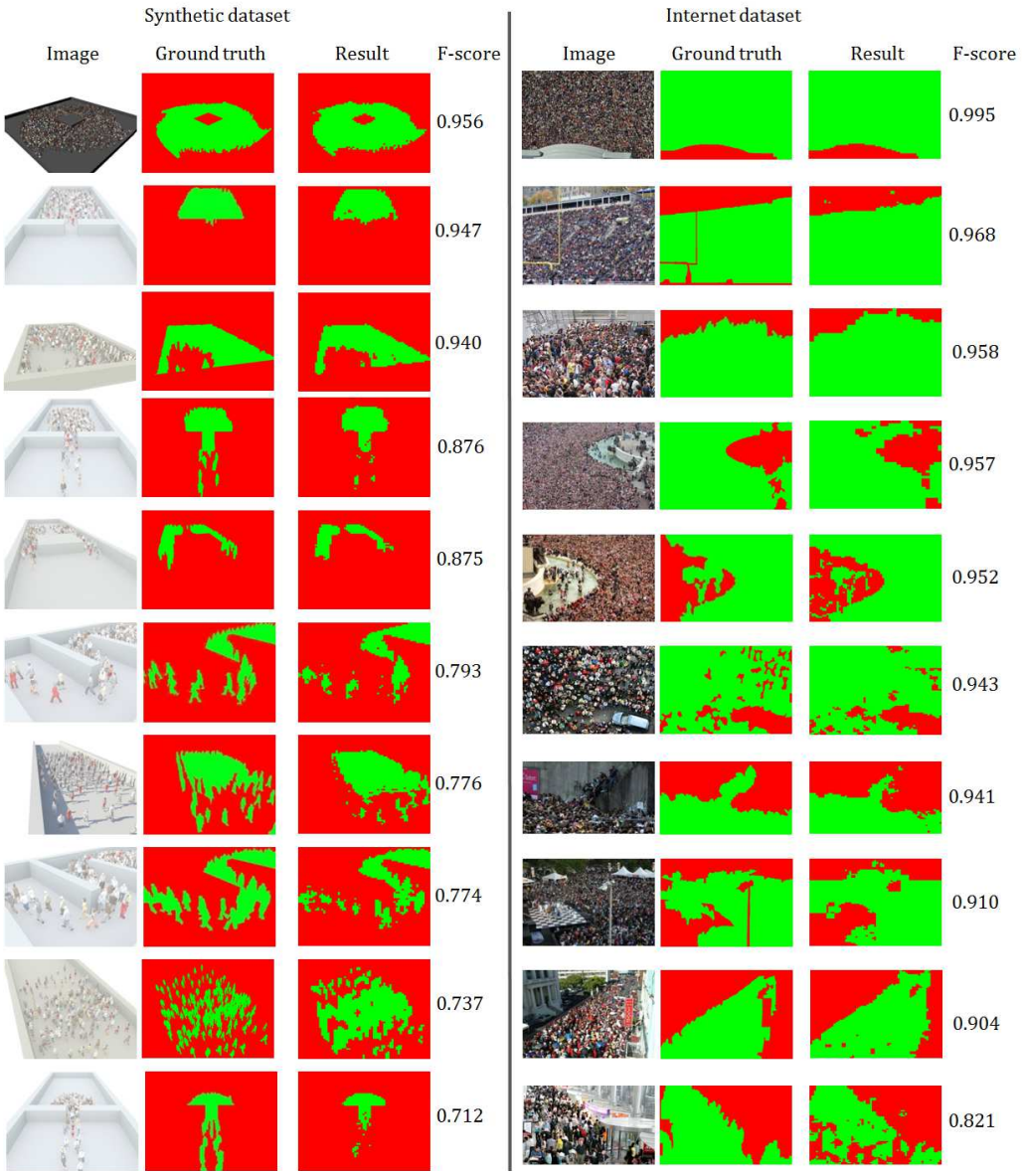


Figure 4: More results and performances: on the left the synthetic dataset, on the right the dataset constituted with images taken from Google Images.

front of them? The result shown on Figure 5a indicates that our algorithm considers that both the inter-individual spaces as well as the people sitting at the terrace of a cafe are part of the crowd.

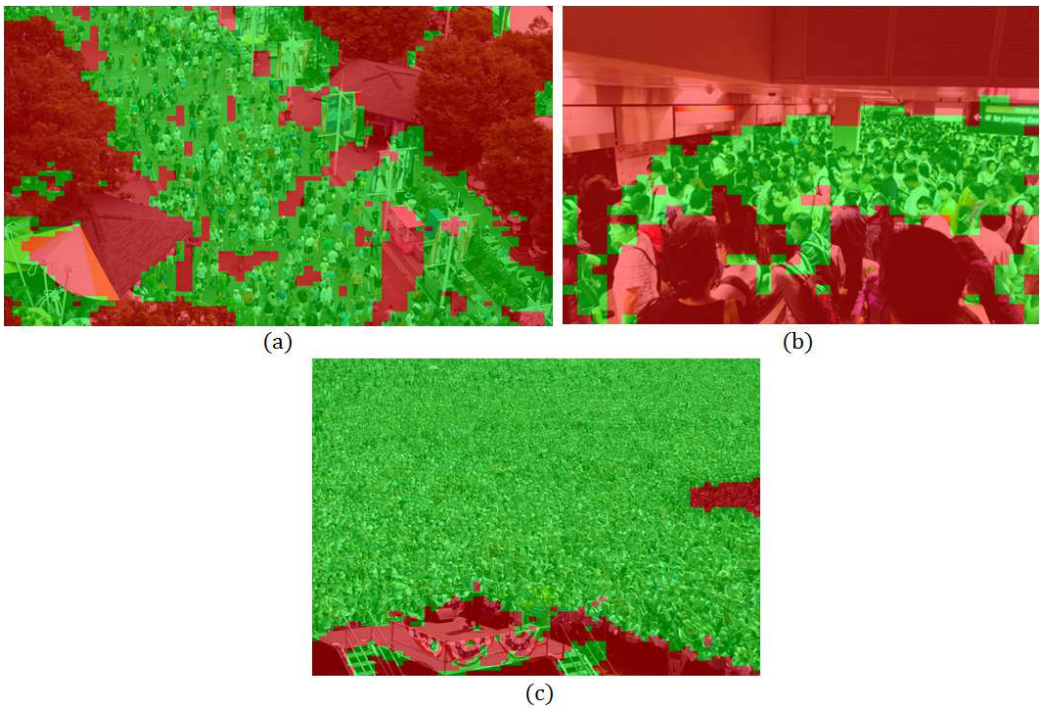


Figure 5: Example of the limitations encountered by our algorithm.

Moreover, as noted in 4.4, the background has an influence on the performances of the algorithm. It has been shown that when the background has a texture too chaotic, the algorithm fails to segment the crowd properly and tends to allocate elements of the background to the crowd class, as for example colorful flags or signboards. Conversely, elements of the crowd can be confused and sorted into the background class if their texture is similar to a background texture. These two cases are illustrated on Figure 5a and 5c.

The entanglement of beams combined with the steps at the bottom of Figure 5c or the furniture present on the right side of Figure 5a show a structural information too close to the crowd for our algorithm. The shadow on the right part of Figure 5c tricks it too. Experience shows that with a higher α this last problem disappears.

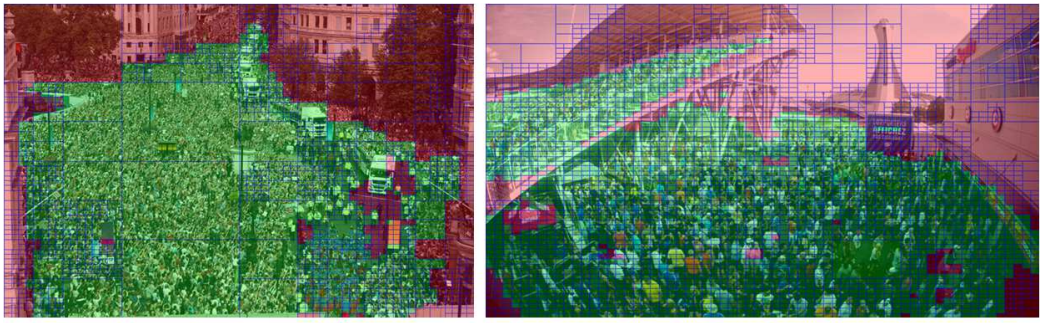
Finally, another limitation of our algorithm is displayed on Figure 5b. The image was shot in a train station with a low-positioned camera resulting in an almost horizontal field of view. As a result, people in the foreground appear much bigger than those in the background. Our algorithm detects successfully the crowd except for the heads closer to the camera which are classified as part of the background. This is due to the fact that they have the same texture as background objects.

5. Conclusion

In this paper, we have combined three kinds of features, extracted at different scales of observation, in order to build a high dimensional multiscale feature vector for each pixel of the image. To separate these multiscale feature vectors into two classes, we have used the diffusion distance instead of the traditional Euclidean distance because we wanted to consider the length and the density of the path between our data. Finally, to optimize the time and volume of computation, we have explored a new technique of coarse-graining using a quadtree. With the combination of these different blocks, we are providing a new and fully unsupervised crowd detection and location algorithm. To conclude our paper, we would like to point out some interesting directions of research for any future work on this method.

First, it is reasonable to think that the three types of features that we are using so far are not the only ones that are relevant for the targeted purpose. Therefore, our multiscale feature vector could be enriched with some new features. We have focused in this paper on static images, however, as it has been reminded in Section 2, motion is also a very important feature of a crowd. We are therefore thinking of including some dynamic features to our multiscale feature vectors.

Second, the way to compute the multiscale feature vector can most certainly be improved in two ways: each type of feature can be explored within its own range of size of windows of observation. Some features are better used locally, some others are more relevant when computed at a larger scale. Moreover, the weight attributed to each type of feature is a point that deserves to be studied furthermore.



(a) (b)

Figure 6: Quadtree and segmentation superimposed: one can see how the algorithm approximates the crowd area.

Finally, we believe that if a human-being is able to detect and locate a crowd in an image in spite of its resemblance to other natural phenomenon,

it is because of his capacity to extract from this image some higher-level information. Raising the problem at a semantic level would provide us with a context that could help us quantify the probability of dealing with a crowd or not.

References

[1] B. Zhan, D. Monekosso, P. Remagnino, S. Velastin, L.-Q. Xu, Crowd analysis: a survey, Machine Vision and Applications 19 (2008) 345–357.

[2] J. C. Silveira Jacques Jr., S. Raupp Musse, C. Rosito Jung, Crowd analysis using computer vision techniques, IEEE Signal Processing Magazine 27 (2010) 66–77.

[3] D. Gerónimo, A. López, A. D. Sappa, Computer vision approaches to pedestrian detection: Visible spectrum survey, Pattern Recognition and Image Analysis 4477 (2007) 547 – 554.

[4] D. M. Gavrila, The visual analysis of human movement: a survey, Computer Vision and Image Understanding 73 (1999) 82 – 98.

[5] B. Leibe, E. Seemann, B. Schiele, Pedestrian detection in crowded scenes, in: Proc. of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, Vol. 1, 2005, pp. 878 – 885.

[6] B. Wu, R. Nevatia, Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors, in:

Proc. of the 10th IEEE International Conference on Computer Vision, ICCV 2005, Vol. 1, 2005, pp. 90 – 97.

[7] Z. Lin, L. S. Davis, D. Doermann, D. Dementhon, Hierarchical part-template matching for human detection and segmentation, in: Proc. of the 11th IEEE International Conference on Computer Vision, 2007, pp. 1 – 8.

[8] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, Vol. 1, IEEE, 2005, pp. 886 – 893.

[9] P. Tu, T. Sebastian, G. Doretto, N. Krahnstoever, J. Rittscher, T. Yu, Unified crowd segmentation, in: Proc. of the 10th European Conference on Computer Vision, ECCV 2008, 2008, pp. 691 – 704.

[10] L. Dong, Fast crowd segmentation using shape indexing, in: Proc. of the 11th International Conference on Computer Vision, ICCV 2007, 2007, pp. 1 – 8.

[11] L. Wang, N. H. C. Yung, Crowd counting and segmentation in visual surveillance, in: Proc. of the 16th IEEE International Conference on Image Processing, ICIP 2009, 2009, pp. 2573 – 2576.

[12] B. A. Boghossian, S. A. Velastin, Motion-based machine vision techniques for the management of large crowds, in: Proc. of the 6th IEEE International Conference on Electronics, Circuits and Systems, 1999, Vol. 2, 1999, pp. 961–964.

[13] P. Reisman, O. Mano, S. Avidan, A. Shashua, Crowd detection in video sequences, in: Proc. of the IEEE Intelligent Vehicles Symposium, 2004, pp. 66 – 71.

[14] V. Rabaud, S. Belongie, Counting crowded moving objects, in: Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 1, 2006, pp. 705 – 711.

[15] S. Ali, M. Shah, A lagrangian particle dynamics approach for crowd flow segmentation and stability analysis, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2007.

[16] M. Manfredi, R. Vezzani, S. Calderara, R. Cucchiara, Detection of crowds gathered in open spaces by texture classification, in: Proc. of the 1st International Workshop on Pattern Recognition and Crowd Analysis, PRCA 2012, 2012, pp. 32 – 35.

[17] M. Rodriguez, I. Laptev, J. Sivic, J.-Y. Audibert, Density-aware person detection and tracking in crowds, in: Proc. of the International Conference on Computer Vision, ICCV 2011, 2011.

[18] N. I. Fisher, Statistical analysis of circular data, Cambridge University Press, 1995.

[19] Z. Farbman, R. Fattal, D. Lischinski, Diffusion maps for edge-aware image editing, in: ACM Transactions on Graphics (TOG), Vol. 29, ACM, 2010, p. 145.

[20] R. R. Coifman, S. Lafon, Diffusion maps, Applied and Computational Harmonic Analysis 21 (1) (2006) 5 – 30.

[21] J. De la Porte, B. Herbst, W. Hereman, S. van der Walt, An introduction to diffusion maps, in: Proceedings International, 2008.

[22] B. Nadler, S. Lafon, R. Coifman, I. G. Kevrekidis, Diffusion maps - a probabilistic interpretation for spectral embedding and clustering algorithms, Principal manifolds for data visualization and dimension reduction 10 (2008) 238 – 260.

[23] C. Fowlkes, S. Belongie, F. Chung, J. Malik, Spectral grouping using the nystrom method, Pattern Analysis and Machine Intelligence, IEEE Transactions on 26 (2) (2004) 214–225.

[24] S. Lafon, A. B. Lee, Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization, Pattern Analysis and Machine Intelligence, IEEE Transactions on 28 (9) (2006) 1393–1403.

[25] Wikipedia, Quadtree, http://en.wikipedia.org/wiki/Quadtree.

[26] P. Allain, N. Courty, T. Corpetti, AGORASET: a dataset for crowd video analysis, in: 1st ICPR International Workshop on Pattern Recognition and Crowd Analysis, Tsukuba, Japan, 2012.